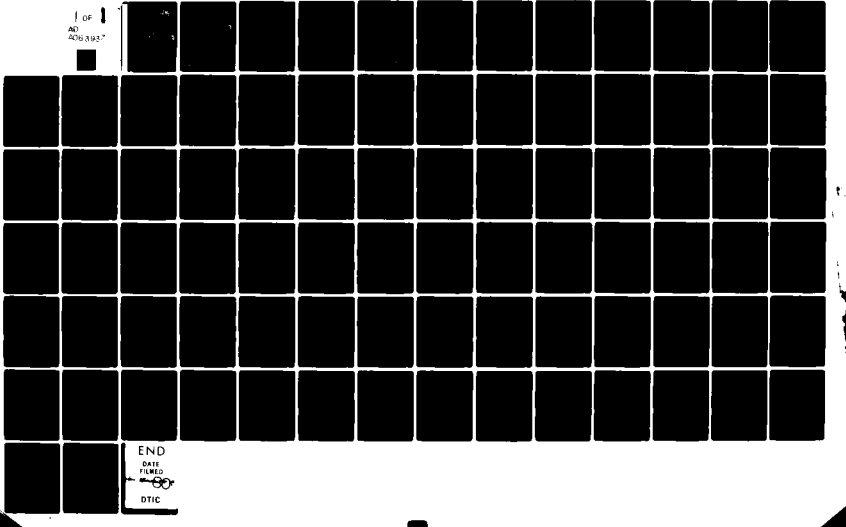
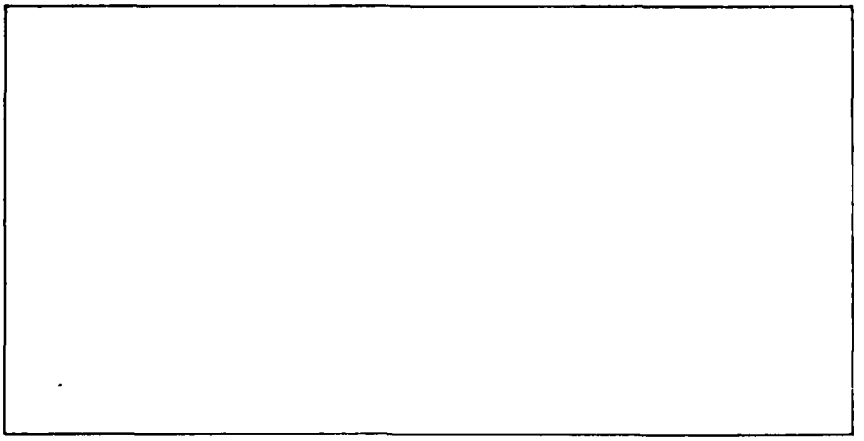
AD-A083 937 SCIENCE APPLICATIONS INC ENGLEWOOD CO F/G 9/2
STUDIES IN PLAN CONSTRUCTION I: ANALYSIS OF AN EXTENDED PROTOCO--ETC(U)
MAR 80 M R ATWOOD, R JEFFRIES N00014-78-C-0165
UNCLASSIFIED SAI-80-028-DEN NL



(12)

LEVEL II



SCIENCE APPLICATIONS INCORPORATED

(12)

STUDIES IN PLAN CONSTRUCTION I: ANALYSIS OF AN EXTENDED PROTOCOL

Technical Report
SAI-80-028-DEN

March 1980

Michael E. Atwood
Science Applications, Inc.

Robin Jeffries
University of Colorado

This research was sponsored by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract No. N00014-78-C-0165, Contract Authority Identification Number, NR157-414.

Approved for public release; distribution unlimited.

Science Applications, Inc.
40 Denver Technological Center West, 7935 East Prentice Avenue, Englewood, Colorado 80111, 303/773-6900
Other SAI Offices: Albuquerque, Ann Arbor, Arlington, Atlanta, Boston, Chicago, Huntsville, La Jolla, Los Angeles, McLean, Palo Alto, Santa Barbara, Sunnyvale, and Tucson

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report Number 2	2. GOVT ACCESSION NO. AD-A083 937	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Studies in Plan Construction I: Analysis of an Extended Protocol.		5. TYPE OF REPORT & PERIOD COVERED Technical Report.
		6. PERFORMING ORG. REPORT NUMBER SAI-80-028-DEN, TR-2
7. AUTHOR(s) Michael E. Atwood and Robin Jeffries		8. CONTRACT OR GRANT NUMBER(s) N00014-78-C-0165
9. PERFORMING ORGANIZATION NAME AND ADDRESS Science Applications, Inc. 7935 E. Prentice Avenue Englewood, CO 80111		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR157-414
11. CONTROLLING OFFICE NAME AND ADDRESS Personnel & Training Research Programs Office of Naval Research Arlington, VA 22217		12. REPORT DATE March 1980 13. NUMBER OF PAGES 65
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Problem solving, planning, computer programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the analyses of long, thinking-aloud protocols collected from two experienced software designers during their initial attempts at solving a moderately difficult design problem. Our method of analysis involves transforming the protocols into a series of rules in a condition-action format. These rules reflect the "policies", or general strategies that guide design activities, "goals", that indicate the actions that a designer intends to accomplish, and "notes", that represent problem-specific information that was generated or retrieved during the course of problem solving. (cont.)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

392 178

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Solving a design problem involves the identification of the components that must be included in a design and the solutions of these components. The identified components represent the subproblems that must be solved in order to complete the design, and these solutions involve the creation and satisfaction of goals. In this report, we focus on the form and content of these goal structures and the manner in which they are generated. We conclude that the design components tend to be enumerated breadth-first on certain classes of design problems, and that the associated goals are explored depth-first. This cycle proceeds through multiple iterations and other processes constrain the expansions of these top-down processes.

PREFACE

This report describes an investigation of the problem solving processes observed in a design task. The discussions presented here are based on a detailed analysis of long, thinking-aloud protocols. Detailed descriptions of the results of these analyses are contained in an appendix, which may be obtained from the authors.

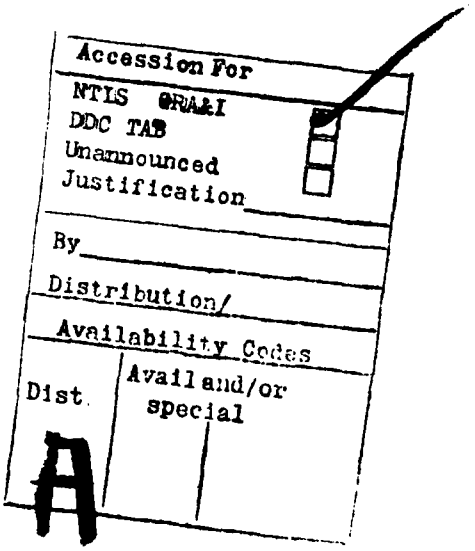


TABLE OF CONTENTS

APPENDICES

(presented in a separate volume)

INTRODUCTION

This report describes an investigation of the problem solving processes employed in a design task. In this context, we will describe a method for analyzing thinking-aloud protocol data and the results of such an analysis.

In discussing ill-structured problems, Simon (1973, pg. 190) notes that "the whole [architectural] design, then, begins to acquire structure by being decomposed into various problems of component design, and by evoking, as the design progresses, all kinds of requirements to be applied in testing the design of its components. During any given short period of time, the architect will find himself working on a problem which, perhaps beginning in an ill structured state, soon converts itself through evocation from memory into a well structured problem."

Design involves the identification of design components and, then, the solution of these components. The identified components represent subproblems that must be solved in order to complete the design, and these solutions involve the creation and satisfaction of goals. The form and content of these goal structures and the manner in which they are generated, therefore, provides a record of the problem solving actions taken by a designer. In this report, the major focus will be on such goal structures.

The problem solving processes involved in design can be described as the creation of goal structures. The second focus of this report is to describe these processes. Simon (pp. 190-191) equates the processes involved in attaining solutions to identified design components to GPS (General Problem Solver; Ernst and Newell, 1969) and those involved with identification to a "retrieval system, which modifies the problem space by evoking from long-term memory new constraints, new subgoals, and new generators for design alternatives." In this report, we will describe the results of these processes and consider the "constraints" and "generators" that underlie these actions and the manner in which these actions are applied.

This report focuses on the goal structures created during the performance of a design task and on the derivation of these structures. We present a detailed look at a protocol collected in a software design task and a brief look at a second protocol. The organization of this paper is as follows: we briefly describe the task of software design. Next, we describe the method of analysis that leads to the identification and definition of these goal structures and associated problem solving processes. We present an overview of the goal structures and processes observed in the protocols and compare these observations with those of others who have investigated behavior in complex tasks. We examine one protocol and, then, give specific illustrations of the designer's problem solving processes and associated goal structures. This analysis is compared to that of a second protocol. Finally, we consider the implications of our results.

TASK DOMAIN AND PROBLEM DESCRIPTION

Software design is the process of translating a set of task requirements (functional specifications) into a structured description of a computer system that will perform the task. There are three major aspects to this description. The original specifications are decomposed into a collection of modules, or substructures, each of which satisfies part of the original problem description. This is often referred to as modular decomposition of the problem. In addition, these modules must communicate in some way. The designer must specify the interrelationships and interactions among the modules. This includes the control structure, which indicates which modules are used by a given superordinate module to do its work and the conditions under which they are used. Lastly, a design may include a definition of the data structures required.

One can think of the functional specifications as indicating the properties that are desired in the final result. The design identifies the components, or modules, that can satisfy these properties. That is, a design delineates what must be done in order to meet the functional specifications. How these modules are to be implemented is a programming task, which follows the design task.

The particular problem to be discussed in this paper is to design a page-keyed indexing system. The problem specifications are shown in Figure 1. This problem was chosen because it is of moderate difficulty,

understandable to individuals with a wide range of knowledge of software design, while not requiring knowledge of highly specialized techniques that would be outside the competence of some expert subjects. That is, a reasonable design could be constructed for this task using only the techniques taught in upper-division undergraduate courses in computer science or those contained in standard textbooks on computer science algorithms. A variety of approaches, however, could be taken to design such a system. A designer must select an approach that can be implemented, identify the algorithms that must be included, and specify how these algorithms will interrelate.

METHOD OF ANALYSIS

We collected thinking-aloud protocols from two experienced designers who were given the page-keyed indexing problem. Both protocols were approximately three hours in length and were very rich in information about design processes. We analyzed the protocols by transforming them into collections of rules in the form of condition-action pairs.

The motivation behind our recoding of the protocol was two fold. Primarily, we wanted a transformation of the original data that eliminated irrelevant material and made as explicit as possible the motivating conditions for decisions and the relationships among facts. We also viewed this process as a first step toward an eventual computer model of the design task. The recoded protocol should serve as the basis for one component of a system that can solve problems similar to the specific problem to be reported here.

These considerations led us to translate the relevant portions of the protocol into a set of condition-action rules. Each rule is intended to represent a decision made by the subject and the features of the problem situation that motivated that decision. We tried to stay as close as possible to the protocol itself, but motivating conditions, and occasionally entire rules, were inferred as necessary to keep the set of rules reasonably complete and consistent.

The first step in our protocol analysis is to partition the protocol into segments that can be ignored and those that will be translated into rules. The ignored segments include digressions (e.g., on the historical relationship between relocatable loaders and modular programming), reviews of the current state of the design, definitions of terms, and unintelligible phrases. A list of the lines that were ignored and a brief description of the topic of each is shown in Appendix B; examples of such passages are given in Figure 2.

Two persons did the encoding into rules, each modifying the rules proposed by the other iteratively, until they both agreed on a set of rules that seemed to capture the protocol. An attempt was made to have rules encompass information at a similar level of detail (as defined by the encoders' intuitions), but no restrictions were placed on the number of lines of the protocol that went into a single rule.

The rules derived from the protocol are contained in Appendix C. To provide the reader with an idea of the correspondence between the actual contents of the protocol and our encoding into rules, some examples are presented in Figure 3.

Each rule is a condition-action pair, written in an IF...THEN format. The left hand side of the rule contains one or more clauses that are true of the current situation and which lead to the generation of the information on the right hand side. The right hand side clauses either set "policies", create "goals" or "note" information in working memory or some external memory. The conditions on the left hand side

PAGE-KEYED INDEXING SYSTEM

BACKGROUND.

A BOOK PUBLISHER REQUIRES A SYSTEM TO PRODUCE A PAGE-KEYED INDEX. THIS SYSTEM WILL ACCEPT AS INPUT THE SOURCE TEXT OF A BOOK AND PRODUCE AS OUTPUT A LIST OF SPECIFIED INDEX TERMS AND THE PAGE NUMBERS ON WHICH EACH INDEX TERM APPEARS. THIS SYSTEM IS TO OPERATE IN A BATCH MODE.

DESIGN TASK.

YOU ARE TO DESIGN A SYSTEM TO PRODUCE A PAGE-KEYED INDEX. THE SOURCE FILE FOR EACH BOOK TO BE INDEXED IS AN ASCII FILE RESIDING ON DISK. PAGE NUMBERS WILL BE INDICATED ON A LINE IN THE FORM /*NNNN WHERE /* ARE MARKER CHARACTERS USED TO IDENTIFY THE OCCURRENCE OF PAGE NUMBERS AND NNNN IS THE PAGE NUMBER.

THE PAGE NUMBER WILL APPEAR AFTER A BLOCK OF TEXT THAT COMPRISES THE BODY OF THE PAGE. NORMALLY, A PAGE CONTAINS ENOUGH INFORMATION TO FILL AN 8 1/2 X 11 INCH PAGE. WORDS ARE DELIMITED BY THE FOLLOWING CHARACTERS: SPACE, PERIOD, COMMA, SEMI-COLON, COLON, CARRIAGE-RETURN, QUESTION MARK, QUOTE, DOUBLE QUOTE, EXCLAMATION POINT, AND LINE-FEED. WORDS AT THE END OF A LINE MAY BE HYPHENATED AND CONTINUED ON THE FOLLOWING LINE BUT WORDS WILL NOT BE CONTINUED ACROSS PAGE BOUNDARIES.

A TERM FILE, CONTAINING A LIST OF TERMS TO BE INDEXED, WILL BE READ FROM A CARD READER. THE TERM FILE CONTAINS ONE TERM PER LINE, WHERE A TERM IS TO 5 WORDS LONG.

THE SYSTEM SHOULD READ THE SOURCE FILES AND TERM FILES AND FIND ALL OCCURRENCES OF EACH TERM TO BE INDEXED. THE OUTPUT SHOULD CONTAIN THE INDEX TERMS LISTED ALPHABETICALLY WITH THE PAGE NUMBERS FOLLOWING EACH TERM IN NUMERICAL ORDER.

A NULL SOURCE FILE INDICATES THAT PROCESSING IS COMPLETED. ERROR MESSAGES AND A TERMINATION MESSAGE SHOULD BE WRITTEN TO THE OPERATOR'S CONSOLE. EACH COMPLETED INDEX IS TO BE STORED ON DISK FOR LATER LISTING.

Figure 1. The Page-Keyed Indexing Problem

(lines 87-95)
this is - also the, this sort of method came out when uh modular program, the idea of module programming - my heavens, might have more than one sub-routine, uh came out. Of course, that came out when - with uh relocatable loaders, which uh - you really need to have that kind of thing with lots and lots of subroutines. Um so I usually try to do things from that point of view, top-down, "let George do it".

(lines 414-418)
Well, computer words, I guess, at this point, well let's not consider that so much as say it's words. (That's certainly a misused word.) Uh, units - whatever unit is necessary to be the length of a pointer.

(lines 523-542)
(Funny-looking box). I didn't really make the boxes right here. Uh I don't know what shape this, these, these things are called, But way back when IBM first came out with their templates, they had a box shaped, shaped like that on it, (whoops, this is supposed to be all, all straight lines here) and that box was supposed to mean, uh, well there are lots of schools of thought, one school of thought was that this box implied a subroutine, and I've always used that, some - and sometimes these boxes imply subroutines, but that's sort of, they're sort of I/O subroutines and you don't really specifying them. At this poinw you're what you're saying - something like, uh the equivalent of saying a FORTRAN read, and this is certainly a subroutine, and yet you aren't really worried much of anything about it, and you have no, probably no control if anything goes wrong either, which is unfortunate, but -. So, that's what boxes that approximate that shape mean.

(lines 957-962)

If there is no next term word, that's unequal because there was a text so we have compared everything. And, if there is, just carry on.

Figure 2. Subject S3 - Examples of Segments Not Encoded

(lines 275-295)
Now at this point, terms, we can't, well we could, uh, it doesn't specify, no, that would just be a fancy sort of thing, would be to try to find terms that are really the same. Well, no. No, you wouldn't want to do that because maybe they might appear in the text. You might actually want to have variations on terms that are really the same, that are actually little spelling differences. Since they're supposed to come out alphabetically, and I can't really think of any better way of doing it, uh, I would want to consider that having read all of the terms, that one would then, could then alphabetize the terms, unless you could have a - some - glean that some terms were much more likely than other terms in which case you would want to put them in that order, wha_ever that order might be.

IF (policy is to look for anomolous conditions) AND (goal to build term table) THEN (note that some terms may be duplicates) AND (create goal to consider eliminating duplicate terms)

IF (goal to consider eliminating duplicate terms) THEN (create goal to determine if duplicate terms are redundant)

IF (goal to determine if duplicate terms are redundant) AND (policy to critique decisions by considering alternatives) THEN (note that duplicate terms may be permitted)

IF (goal is to order terms) AND (specifications say output of terms is to be in alpha-order) THEN (note that term table can be constructed by reading in terms and alphabetizing)

IF (goal is to order terms) AND (policy is to be efficient) AND (best order of terms is on basis of frequency of occurrence) THEN (note that frequency order for terms is better than alpha-order)

(lines 408-413)
While I'm thinking of trying to make a list, that - but that there must be a pointer to each list, or, well a pointer to each list and fixing it to be a certain number of pointers, such that the uh, pointers themselves are actually in a vector of NT units or words,

IF (goal to build table of lists) AND (noted to associate a list with each term) THEN (create goal to have a pointer to each term (list))

IF (goal to have pointer to each term (list)) AND (there are NT terms) THEN (note that there are NT pointers) AND (note that there will be a pointer to each term) AND (note that pointers will be stored in a vector)

Figure 3. Subject S3 -- Examples of Encoded Segments

must consist of some member(s) of the set of currently active goals or policies, or of facts (notes) retrieved from long-term, working, or external memory.

Policies are general strategies that a particular designer uses in most or all design tasks. While certain attributes of a problem may cause a given policy to be invoked, the policy itself is problem-independent. Policies represent a designer's approach to design. They influence the direction that the design will go, in that when the subject has a choice among several alternatives, the one chosen will be consistent with his or her policies. Policies are frequently found among the left hand side clauses of rules, as they motivate many of the design decisions the subject makes. We have assumed that all policies are activated at the beginning of the problem solving session. We show this by simply listing at the beginning of the encoding a set of rules, each with a policy on its right-hand side and simply "start of problem" on its condition side.

Goals indicate activities that the subject intends to accomplish. Although goals name what needs to be done, they do not specify how things are actually to be accomplished. They represent the subject's internal representation of the current status of the solution -- what is being worked on, what remains to be done. Some goals can be satisfied almost as soon as they are created, while others invoke subgoals, which may, in turn, invoke their own subgoals.

Notes, the third element found in our rules, include all information specific to the problem that the subject either generates or retrieves while expanding the design. Some of this information is derived directly from the problem specifications or the developing design. Other information comes from known computer science facts or concepts. In addition, a designer makes a large number of inferences during the course of constructing a design, and we have encoded the results of these inferences as notes. Notes include information that is recorded in or retrieved from either the subject's internal memory or from external memory.

From the appendices and the examples shown in Figure 3, the reader will note that our encodings are quite specific. No attempt has yet been made to aggregate the rules into more general constructs. This was done in order to keep rules closely tied to particular segments of the protocol. In this regard, the rules can be considered a translation of the protocol into a form that is easier to present and deal with.

We also intend these rules to be the basis for a model of the processes involved in software design. A great many components of the solution process will need to be specified in much more detail before such a model can be developed. In particular, a major component which is absent from this analysis is the executive process that interprets these rules. However, the rules were constructed with some assumptions in mind about how this interpreter would operate.

The executive's main functions are to: 1) choose which among the potentially relevant rules is to fire; 2) manage information that is recorded in memory and retrieve facts and inferences as needed; and 3) keep track of which goals are active and when goals are satisfied. We have nothing to say about choosing among rules, since we only see evidence of the rule that is actually selected in the protocol.

Much of the second function, managing memory, will depend on the memory representation chosen. We have assumed that information gets retrieved from long term memory or stored in working memory whenever necessary. Also, the executive can make simple inferences about memory contents, such as "there is no information in working memory on topic A", or "all cases of type B have been considered", or "C has recently been recorded on external memory".

Aspects of the third function, keeping track of active goals, are implicit in the way the rules are written. We assume that goals are hierarchically organized, although not necessarily into a single, unified hierarchy. Goals are satisfied when the executive determines that their purpose has been accomplished, frequently by the recording of one or more notes. A goal may generate subordinate goals to accomplish its purpose; in that case, the executive temporarily suspends work on the original goal until the subordinate is achieved. Not all goals are satisfied; if achieving a goal appears impossible or unprofitable, it can be deferred or abandoned. In our encoding of the protocol, we have assumed an executive that will handle these tasks so

that they are transparent; therefore, we have not made explicit reference to them in the rules.

This method of analysis produces a local encoding of the protocols and makes no assumptions about the global problem solving processes employed by designers. Further, the properties we attribute to the executive do not dictate the overall structures of the design processes. Only inferences necessary for local connectivity between rules were made in these encodings. What emerged from this translation was the goal structures that represent global properties of the protocols.

GOAL STRUCTURES: AN OVERVIEW

There are two sources of evidence for the problem solving processes used by our expert designers. Components are the elements included in each iteration of the design, as recorded by the subject in external memory. The goals, as captured in our encoding of the protocols, trace the processes that lead to the definition and decomposition of the components. The overall structure of the problem solving process is hierarchical; that is, it proceeds from abstract to detailed representations of the design.

In the protocols, there are multiple iterations of an identification-solution cycle. Initially, subjects produce abstract designs. In each successive iteration, the design is expanded to include more details. Each iteration produces its own goal structure, and each ends with the satisfaction of all of its goals.

The initial iteration is based on a very abstract decomposition of the original problem. This decomposition, which we term the "problem model", appears to be relatively independent of a given subject's level of expertise, in the problem used here. A problem model identifies the major components that will be required to reach a solution.

The fact that such a problem model was easily derived and that its basic content did not vary significantly as a function of experience, qualifies the conclusions that can be drawn from this research. In other design tasks (e.g., the design of compilers or artificial

intelligence systems), where the development of a problem model may require more of a designer's resources, different problem solving processes may occur. That all subjects can correctly and easily parse the indexer problem into its constituent parts, may lead to the iterative design process we have observed.

The initial iteration, which is based on the problem model, leads to the formulation of the abstract design. Further iterations expand the components of the abstract design to greater levels of detail. The definition of each component is based on the generation and satisfaction of goals, which successively refine the designer's understanding of that component.

At a macro-level, the level at which components are added to the developing solution, our subjects proceed in a top-down, breadth-first manner through multiple iterations. As a result of this breadth-first mode of expansion, the subproblems and associated goals involved in the design are identified.

At the micro-level at which these goals are expanded, however, solutions are explored depth-first. That is, once a subproblem is identified, there is a strong tendency to resolve it before the next subproblem is considered. These depth-first expansions are also guided by policies, which limit the number of alternatives to be considered, and by information that is generated to describe the functions that the solution must satisfy. A given component may be considered in more than one iteration, with more detailed consideration given in each

successive cycle. At each iteration, however, the depth-first expansion continues only until a level of detail "appropriate" for that iteration is attained. For example, the initial iteration may consider the functions that must be performed, while the following iteration would consider mechanisms to perform these functions, followed by consideration of the implementation of these mechanisms. These "stopping rules" are also part of the processes that guide the generation of the goal structures.

In a complex task, the top-down processes described above must interact with other processes that constrain the alternatives that the top-down processes generate for inclusion in a solution. This is one function of the policies mentioned above, since these affect the generation and evaluation of alternatives. They may, for example, indicate which of several alternatives is preferable or cause other solutions to the current subproblem to be generated.

These constraining processes are also apparent in the generation of information that, although it is not immediately useful, is potentially relevant later in the problem solving process. Some of this information is, in fact, irrelevant to the solution, and after it is once generated, it is ignored, or at least never explicitly retrieved. Information that later becomes relevant, however, may be retrieved long after it was generated.

In summary, the top-down processes consist of the breadth-first expansion of the components and the depth-first explorations of these components, and this is restricted by other processes.

Related Research

Here, we will briefly compare the processes underlying the generation of designs with those reported by other researchers who have investigated the solution of complex tasks. In general, these studies also reveal the basic top-down processes described above. They do not, however, clearly indicate the alternation between breadth-first and depth-first expansions or the type of processes we have observed to constrain the top-down expansions.

Bhaskar and Simon (1977) looked at how an expert solved thermodynamics problems. They found that their problem solver consistently worked top-down, but that the problem was solved breadth-first (i.e., iteratively) only if it surpassed some threshold of difficulty. Their model, SAPA, focuses on the solution of the well-structured problem produced once the correct form of the general energy equation has been retrieved. They also show that the equation was retrieved according to a standard template, and that processes that examined key words in the problem description constrained the initial equation by dictating the inclusion or deletion of certain terms.

Greeno, Magone, and Chaiklin (1979) describe how geometry problems, particularly those involving constructions, can be solved by a model called Perdix. When Perdix cannot solve a problem directly, it tries to instantiate a "plan-schema". A plan-schema "includes information about global features of the situation that are required for the plan to be feasible; for example, the plan for proving two

triangles congruent requires that two triangles be present" (pg. 8). This schema is activated if the features of the problem situation can be made to meet certain prerequisites. If no plan-schema is immediately applicable, auxiliary lines (constructions) can be drawn to satisfy the missing prerequisite of an "almost" applicable schema. That is, this "planning system produces constructions by a process of partial pattern matching and pattern completion. Perdix recognizes that a situation partially matches a pattern that satisfies the prerequisites for a plan-schema and then activates productions that complete the required pattern" (pg. 8). Once the appropriate plan-schema is decided upon, the model proceeds to solve the problem in a purely top-down manner.

While Perdix solves geometry problem completely in a goal-driven manner, Greeno et al. note that their subjects occasionally used a working-forward strategy. This strategy is similar to our discussion of the generation of information that may be relevant only later in the problem solving process that was discussed above. A line would be drawn or an inference made that was not required by the current subgoal or plan-schema. This information could be used later in the solution to instantiate a new plan-schema. In its present form, Perdix cannot account for such behavior, and Greeno et al (pg. 36) suggest that the basic problem is in formulating the control processes that keep the generation of such information "under control".

An expert solving mechanics problems in physics (Larkin, 1977)

also used a top-down, breadth-first approach. After deriving the relevant features of the situation from the problem description and a diagram, Larkin's expert proceeded to solve an abstracted version of the problem and then used this abstract plan to generate a detailed solution. Larkin models this behavior with a purely top-down, hierarchical system that, for the most part, finesses how the appropriate plan is decided on, and deals primarily with how the problem is solved within the constraints of the plan.

There are two aspects that are common to these projects and to similar efforts (see Chase and Chi (1979) for a review). First, identification of the known components underlying the problems investigated is a process of pattern recognition. That is, problem solvers have in long-term memory some number of patterns that correspond to known problems, and some features of the problem cause one of these to be retrieved. Second, and perhaps as a result of the first point, all these studies focus more on the solution of well-structured problems rather than on their selection.

Thus, other researchers analyzing skilled problem solving in semantically-rich domains have found that, on non-trivial tasks at least, good problem solvers use a top-down, breadth-first expansion of a solution. The rk surveyed above, however, has not shown the depth-first exploration of subgoals or the types of processes that constrain these top-down expansions we have found here. We attribute this both to differences in the level of analysis and to the nature of the problems being solved.

A BRIEF OVERVIEW OF THE PROTOCOL

We gave the page-keyed indexing problem to an experienced computer systems analyst and asked her to think out loud while generating a solution. She worked on the problem for over two and a half hours, producing almost thirty single-spaced pages of remarks. Because of the extreme length of the problem solving session, we will present here only a brief summary of the protocol. A more detailed overview is contained in Appendix A. Along with her verbal remarks, the subject produced a flowchart. Close examination of this document, reproduced as Figure 4, should make the final design solution reasonably clear.

The subject, S3, has degrees in physics and electrical engineering and about 15 years programming and design experience. Although she is aware of the current literature on software design methodologies, S3 has no formal training in this area. Nor has she designed any type of indexing system.

The protocol consists of two major iterations toward a solution of the indexer problem. In the first third of the session, S3 generates a "high-level flowchart", a very abstract solution to the problem. The remainder of her time is spent expanding the boxes of this flowchart to a greater level of detail, and we have divided this iteration into two segments.

We will briefly describe these three segments of S3's protocol. Notice that each segment begins with the identification of the

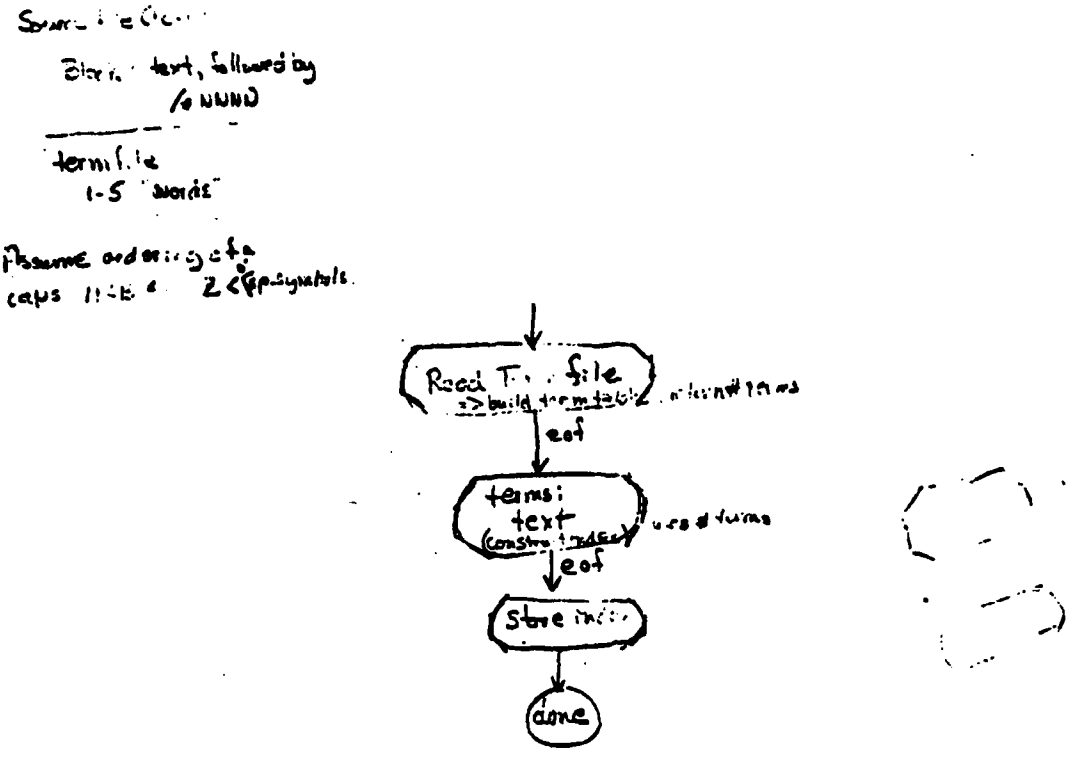


Figure 4. Subject S3 -- Design

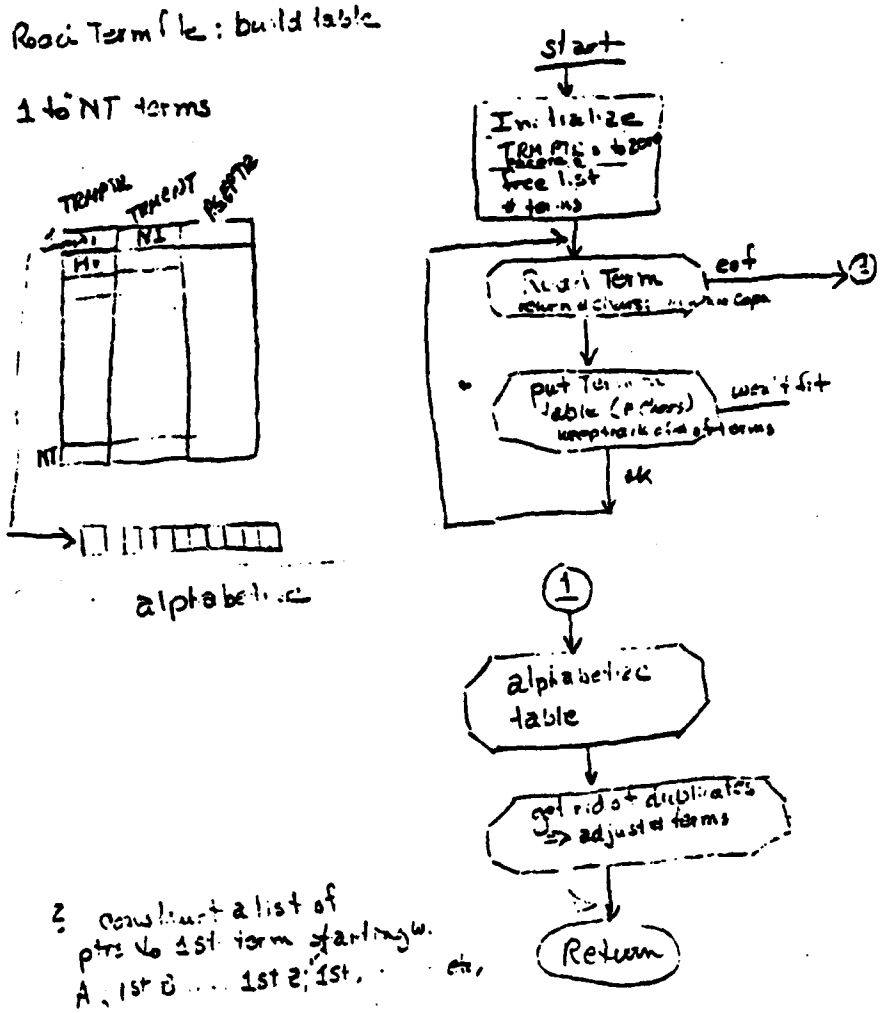


Figure 4. Subject S3 -- Design
(continued)

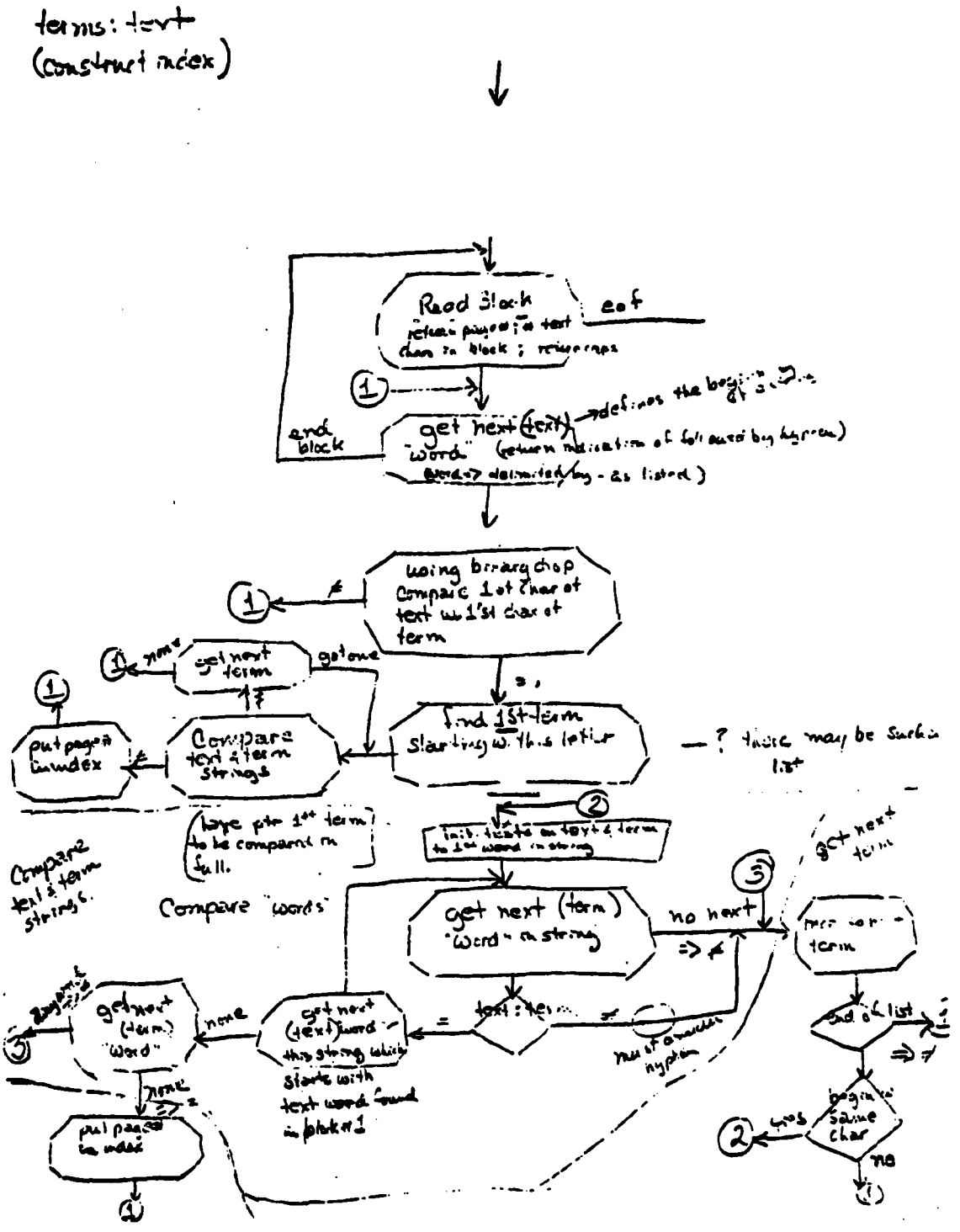


Figure 4. Subject S3 -- Design (continued)

functions that must be performed and, then, detailed consideration of each of these functions.

The initial segment of the protocol, which leads to the construction of the "high-level flowchart", or abstract design, is based on S3's inferred problem model, that was developed very soon after reading the problem description. This problem model indicates that the form of the solution is to "read the term file in order to build a term table, compare terms to text, and store the index."

Although all components of the problem model were considered in the initial segment, the most consideration is given to the structure of the input files and the term table. This problem involves two input files (the term file and the text file), and each is considered in turn.

The only complication that occurs with the text file is that the specifications are ambiguous as to its format. Policies are used to evaluate the two alternative formats, with the final selection based on the ease with which errors in this file could be handled.

Consideration of the term file leads S3 to realize that she will need an internal data structure to store the terms and the results of the comparison with the text -- the page numbers. This data structure, which she calls the "term table", is to be a table of linked lists, one for each term. This structure is then considered, and its definition is based on a policy that leads her to minimize the storage requirements involved. She also notices that the terms will have to be

alphabetized and that duplicate terms may occur. No external notation, however, is made of this information.

In considering how the text and terms will be compared, S3 notices that hyphens can serve two functions; they can be a character in a word, and they can separate words at line boundaries. She realizes that this has implications for how the compare routine will operate. Some time is spent trying to decide how to handle the problem of hyphens, but she comes to no resolution and decides to postpone consideration of this issue.

Once the input files have been adequately considered and the hyphen issue is set aside, the abstract solution is developed fairly quickly. As can be seen from the first panel of Figure 4, she intends to "read the term file", compare "text:terms", and "store index".

The remainder of the protocol is an expansion of these boxes. For convenience, we have divided it into two segments. The first of these is concerned with reading the terms and building the term table. The latter primarily concerns the comparison operation. Since the "store index" module was only briefly considered, we have also included it in the final segment.

S3 begins the second segment with a more detailed examination of the term table data structure. She decides that the terms themselves will be stored in a string table; the term table will contain a pointer to the term and a count of the number of characters in the term. The facts that the terms should be alphabetized and that duplicate terms

may occur, which were generated in the first segment, are reconsidered here, and appropriate modules are added to the flowchart.

Each of these aspects of the term table is considered in turn. Although S3 decided earlier that the page numbers in which terms appear would also be stored in this structure, this topic is not considered in this segment.

The remainder of the second segment is concerned with recording on the flowchart the decisions made up to this point. The resulting flowchart is shown in panel 2 of Figure 4.

In the final segment, S3 describes how the actual compare operation is to be performed. In terms of length and the number of modules added to the flowchart, this is the most complex of the three segments and involves the largest goal structure. Each component is considered in detail and independently of the others.

S3 begins with a module that will read in a block (page) of text and extract the page number. At this point she also returns to the unresolved hyphen problem, but decides that the "read block" module is not the place to deal with it.

She next determines that the text and terms should be compared word by word, as opposed to comparing an entire term string to the text. She begins looking for a term that begins with the first letter of the current text word, using a "binary chop" routine. When she realizes that this will not necessarily bring her to the first term starting with that letter, she adds a routine that searches a list of "first terms starting with [each] letter."

Consideration of the compare operation proper focusses on how to determine if the result is a match, a mismatch, or if it is necessary to compare additional words. This leads her to notice a similarity between this compare routine and the task of comparing, or verifying, two files. The analogy is not an accurate one, however. This mistake leads her to assume that the terms and text will only match if they are the same length; that is, the text may not continue beyond the end of the term.

She next decides what she must do once a match or a mismatch is found. The "mismatch" case leads her to either compare the text to additional terms or to advance to the next text word. On a "match", she must store the page number for that match. This causes her to notice that there is not room in her term table for page numbers; she then adds a vector of pointers to linked lists of page numbers. The flowchart produced in this segment is shown in the third panel of Figure 4.

Next, she discusses some of the functions the "store index" module must perform -- e.g., put commas between entries and put headings at the top of each page. She chooses not to expand this module, because it is "uninteresting".

The remainder of the protocol contains a critique of the compare operation and a discussion of how the hyphen problem would be solved. She finishes by discussing with the experimenter what remains to be done to solve the problem, and how she would approach it. Since this

section of the protocol does not involve additions or modifications to the flowchart, we will not consider it, in detail, in this analysis.

RESULTS OF THE ANALYSIS

In this section, we present some of the results obtained from S3's design and from our encoding of this protocol into rule form. First, we will consider how each of the elements involved in these rules -- policies, goals, and notes -- relate to the development of the goal structures constructed to solve a design problem. Because it is based on these elements, this analysis provides a fairly local view of these processes. In addition, we will consider the actions of these processes from a more global view. This involves errors in the design, frequently recurring topics, and other aspects that influence or result from these processes.

We defined policies as general strategies that represent a designer's approach to a design task and guide the manner in which the design will be expanded. As such, they provide fairly direct evidence of the problem solving processes employed in a design task. Goals, on the other hand, more closely represent the results of applying these processes, and notes represent the information that these processes use to make decisions and, perhaps, information that they cause to be generated.

Policies

The primary function of policies is to guide the generation and evaluation of design alternatives, to prescribe functions that the design components must satisfy, and to aid in determining which aspect

be efficient

choose the most convenient of multple alternatives

critique decisions by considering alternative solutions

defer goals that represent tricky problems

define returns from modules

divide problem into inputs, processes, and outputs

expand top-down

look for anomalous conditions

resolve ambiguities by asking user/experimenter

use flowcharts

Figure 5. Subject S3 -- Policies

of the design should be considered at a given time. The policies identified in our encoding of S3's protocol are shown in Figure 5.

The primary policy concerned with generating alternative solutions is to "critique the design by considering alternative solutions." For example, when considering how to define the results of the text and term comparisons, this policy leads to noting that this could be a list of page numbers on which terms appear as well as a count of the number of times each term appears.

The policies involved in evaluating, or selecting, alternatives include "ask(ing) the user/experimenter" and "choosing the most convenient of multiple alternatives." A selection can also be made on the basis of "efficiency" or by selecting the alternative that involves the fewest "anomalous" conditions. An example of the latter is the selection of a format for the text file that makes errors, or anomalous conditions more easily dealt with.

The policy to "look for anomalous conditions" can also be used to prescribe the functions that a module must perform. For example, when it is noted that terms may contain errors, a test for errors is included in the module that will read the terms. The policy to "divide problem into (initializations), inputs, processes, and outputs" is similar in that each expansion of the design involves consideration of initialization functions.

Some policies aid in determining which components of the design to consider next. For example, if sufficient information does not yet

exist to support detailed consideration, the policy to "defer goals that represent tricky problems" is apparent. The policy to "define returns from modules", which is most apparent in the latter two segments of the protocol, clearly controls selection of the modules to be considered at a given time.

The remaining policy ("to use flowcharts") reflects the chosen documentation medium for the design. It is clearly related to some of the policies described above, and, as a result, may serve multiple functions.

In summary, policies constrain the top-down expansion of a design solution by generating and selecting design alternatives to be incorporated into the design. As a result of these processes, the goals necessary to perform the design task are created.

Goals

Goals, in our encoding, indicate the actions that the designer intends to accomplish to complete a design task. This generation proceeds in an orderly manner. In this subsection, we will consider the generation of goals, as it was captured in our encoding. We will consider both the order in which goals were created and their relations to other goals.

In the course of solving this problem, S3 generates a large number of goals. For each rule, in order of its use, we tabulated which goals were active when the rule was invoked and which goals were satisfied by the invocation of that rule. This results in a "goal stack" that

	active	satisfied
10.	41	
11.		41
12.	10	
13.		10
14.	15	
15.	15, 30	
16.	15, 30, 17, 16	
17.	15, 30, 17, 16, 22	
18.	15, 30, 17, 16, 22, 34	
19.	15, 30, 17, 16, 22, 34, 38	
20.	15, 30, 17, 16, 22	38, 34
22.	15, 30, 17, 16, 22, 11	
24.	15, 30, 17, 16, 22, 11, 4	
26.	15, 30, 17, 16, 22, 11	4
27.	15, 30, 17, 16, 22, 11, 2	
28.	15, 30, 17, 16, 22, 11	2
29.	15, 30, 16	11, 22, 17
30.	15, 30, 16, 40	
31.	15, 30, 16, 40, 31	
32.	15, 30, 16, 40	31
35.	15, 30, 16, 40, 19	
36.	15, 30, 16, 40, 19	
37.	15, 30, 16, 40, 19, 20	
38.	15, 30, 16, 40, 19, 20, 29	
39.	15, 30, 16, 40, 19, 20, 9, 29	
40.	15, 30, 16, 40, 19, 20, 9, 29, 32	
41.	15, 30, 16, 40, 19, 20, 9, 29	32
42.	15, 30, 16, 40, 19, 20, 9, 29, 37	
43.	15, 30, 16, 40, 19, 20, 9, 29	37
44.	15, 30, 16, 40, 19, 9, 29	20
45.	15, 30, 16, 14	40, 9, 14, 29, 19

Figure 6. Subject S3 -- Example of Goal Stack

indicates which goals the designer is considering at a given point in the design process. These goal stacks, one for each of the three segments into which we have divided this protocol, are contained in Appendix D and a portion of one is shown in Figure 6.

Many of the goals generated by S3 are descendants of others. That is, in order to accomplish the functions of some goals, subgoals are created. These relations, which were derived directly from our encoding, are contained in Appendix E and a portion of one is shown in Figure 7.

In considering the goal stacks, notice that the number of goals being considered at a given time varies. At several points, the number of goals becomes relatively small and then begins to increase. At those points where the number of active goals reaches a relative minimum, the protocol indicates a shift of focus on the part of the designer from one aspect of the design to another. In fact, the episodes described in the overview of this protocol (see Appendix A), are delimited by such shifts, with each episode beginning with the generation of a small number of goals and ending with their satisfaction.

S3's problem solving processes, therefore, control both the number and purpose of goals that are active at a given time. This ensures that the goal information, which represents the designer's conception of the current status of the design, remains within working memory capacity.

Comparing the goal stacks with the hierarchical arrangement of

1-41. select language to be used
2-10. create problem model
3-15. describe input files
5-17. describe text file
6-22. determine record structure of text file
7-34. examine known information on text file
8-38. reread specifications for text file
9-11. decide between A and B
10-4. choose the more convenient of A and B
11-2. choose the alternative (A or B) that makes errors or anomalous cases more easily dealt with
5-16. describe term file
21-36. order the terms
22-26. determine the basis for ordering (terms)
29-6. consider other solutions to ordering terms
30-8. consider terms in alpha-order
24-21. determine record structure of term file
25-33. examine known information on term file
23-23. determine size of term file
4-30. draw top level flowchart (FC) of processes

Figure 7. Subject S3 -- Example of Goal Structure
(Note: first number indicates relative order -- second identifies goal in "goal stack")

goals provides evidence of how this management takes place. Within each segment of the protocol, the first few goals, which may be considered to be the "major" goals of that segment, are identified relatively quickly and without overt consideration of alternatives. Each major goal then invokes appropriate subgoals. Once these subgoals are identified, however, their sub-subgoals are generated in a depth-first manner. That is, once a subproblem, at some level of detail, is posed, it is resolved before additional subproblems, that may have been identified earlier, are considered (see Figure 7). In general, episodes are depth-first explorations of a single goal that represent some components of the design that the designer considers to be necessary. This topic-by-topic exploration is one mechanism that is used to ensure that the goal information remains within reasonable bounds.

In summary, goals represent the actions that the designer intends to accomplish and the selection of goals is based, in part, on policies. This selection is also based, however, on information that is generated during the course of problem solving. We have encoded such information as notes.

Notes

Of the elements involved on our rules, notes are by far the most numerous. The primary use of notes appears to be to store information that is relevant to the generation and evaluation of design alternatives. As was indicated above, the problem solving processes that create goal structures have access to this information and, as we

will show below, cause some of it to be generated. The notes involved in our encoding are contained in Appendix F.

Most of the information we have encoded as notes was derived during the course of problem solving and is related to specific goals. That is, the creation of a goal leads to the generation of information that is relevant to the satisfaction of that goal.

There may, however, be alternative methods for satisfying a given goal, and notes can indicate the alternatives, the functions that must be performed, and the relative advantages and disadvantages of each alternative. The goal to "determine (the) record structure of the text file", for example, leads to noting that "text file records are 'blocks'", "page number appears after block of text", "page number may be part of block", and "page number may be separate from block". These last two represent alternative solutions to this goal, and a selection is made on the basis of other information that is generated ("locating page number can be done by scanning backwards from end of block", etc.)

Such information is explicitly generated to guide the development of the goal structures. Other notes, however, were not explicitly generated during the course of problem solving. That is, while most notes initially appear on the right-hand sides of our rules, some inferred information initially appears on left-hand sides. Such information appears to come from several sources, and examples are provided in Figure 8.

The notes that we attribute to the "executive" appear to be

"executive"
- attempts to resolve hyphen problem have failed
- no data structures or variables are yet defined

"flowchart"
- flowchart contains "read term file"
- flowchart contains "end of list" test

"problem specifications"
- problem specifications state that unit of text is block
- output of terms is to be alpha-order

"inferences from problem specifications"
- text block will be a variable length record
- text file is an input file

"computer science"
- results of read are being stored in memory incrementally
- list may be empty

"textbooks and indexing"
- text may be justified
- terms may be within terms

"simulating effects of design"
- text is not equal to term (at this point)
- text word is found (at this point)

"generated"
- assume that characters are lexically ordered
- compare should begin with next text word

Figure 8. Subject S3 -- Examples of Notes

generated whenever there is a need for information on the current state of working memory. Notes derived from "simulating" the effects of a segment of the design serve to regenerate needed information and also aid in determining what returns from modules should do. Other information is derived from S3's knowledge of "computer science" and "textbooks and indexing". When it is needed, relevant information from these sources is retrieved from long-term memory.

Other information is derived from external sources. This includes information recorded externally on the "flowchart" and the "problem specifications". When necessary, however, the control structure can cause "inferences from the problem specifications" to be generated.

In summary, notes are related to specific goals and they contain information that is used in generating goal structures. In this regard, the function of notes is similar to that of policies. Notes are similar to goals in that both represent information about a specific design task. By indicating necessary design functions, notes may lead to the generation of a given goal.

Up to this point, we have been looking at the protocol at a fairly local level, that of the policies, goals, and notes involved in solving this design problem. We conclude that policies control the generation of goals, that goals are generated in a very orderly manner, and that notes represent information that is generated and then used to control the generation of goals. In the remainder of this section, we will consider certain episodes and recurring topics that provide insights, from a more global view, of S3's problem solving processes.

Problem Solving by Debugging Almost-Right Plans

There are several instances in S3's design where information was generated that caused her to retrieve previously developed plans. This type of problem solving has been referred to by Sussman (1977) as the "debugging of almost-right plans." For reasons that are not clear, however, these plans were not completely "debugged" at the time they were incorporated into the design. This is, in part, a consequence of the unfinished state of the design. Had we taken S3 through more iterations of the design, which is the way in which designs are usually generated, she would most likely have discovered and corrected all of these problems. However, the fact that they were produced at this point provides information about the processes that guide S3's design behavior.

The clearest example is S3's assumption that the term must be the same length as the text for a match to occur (lines 947-962). Her model of the compare process is something like: "get a word of text, get a word of term, compare, loop until a failure occurs or everything is compared". At the point in the design where the comparison has succeeded for the first word of a term and some text word, she considers how to determine when everything has been compared. This leads her to noticing an analogy between this compare operation and the process of verifying two files (949-950). Verifying files is an algorithm that is well understood by S3, and she proceeds to apply some cautions that are prescribed by the file verification template. The

primary one of these is to make sure that the files are of equal length. This is inappropriate in the indexing system, where a term is a substring of the text.

This error is an instance of the application of a well-learned, quite rigidly defined knowledge structure to a situation. Some features of the situation led her to realize that this knowledge structure is relevant; in this case, it is apparently the fact that she is considering a compare operation and the termination conditions for this compare. When she activates the knowledge structure, some procedures related to its use are also activated. Such procedures could indicate common errors to be checked for, a particular type of data structure that is needed, etc. In this case, it is the need to check for simultaneous termination of the files. These procedures are apparently executed without any additional checks for appropriateness.

Design Modification Strategies

S3 does not backtrack. When she finds it necessary to modify or expand upon a previously completed component of the design, she consistently makes modifications only within a single module, never at the interfaces to other modules and never makes changes that could propagate across several components of the design.

The first example is her decision to count the terms as they are read in (646-647) and then, when S3 notices that the "get rid of duplicates" module will alter the number of terms, to adjust the count within that module (649-650). Both modules were added to the design

previously and, at that time, no consideration was given to counting the terms. In considering the compare operation, however, the need for a count is noticed. S3 originally decides to have the "read term" module do the counting, because it cycles through the terms. However, the "get rid of duplicates" module must also cycle through the terms, and if it counted the terms, no "adjustment" procedure would be needed. The count of terms is an output of the "read term" module that could potentially be used by modules between "read term" and "get rid of duplicates". Thus, moving the procedure that counts the terms could possibly have interactions elsewhere in the design. While we have no explicit evidence that she makes the decision for these reasons, her modification resulted in the minimal change to the design.

A second example occurs when S3 notices that the planned "binary chop" routine does not guarantee finding the first term starting with the indicated character (832-840). She resolves this by adding a data structure that contains a pointer to the first term starting with each character and a module that searches this data structure for the term she needs (841-857), making the "binary chop" module unnecessary. However, she does not delete the "binary chop" module. Again, there is reluctance to delete a module that has been incorporated into the design.

Although these decisions not to backtrack can be construed as producing inefficiencies in the design, viewed from another perspective, they may actually improve the efficiency of the problem

solving process. This protocol represents the initial iterations of a design; additional iterations and refinements remain to be performed. These modifications are efficient in that by restricting changes to single modules, this initial iteration can proceed more smoothly. This would not be the case if more efficient changes were effected, requiring frequent and extensive changes to the entire design. We have only observed design behavior, however, in the initial iterations. Were we to examine it in later iterations of the design, we would expect to see more attention to modifying design components that were inefficient from a software design perspective. Such attention is apparent in the final section of S3's protocol, where she describes what would happen on succeeding iterations.

Recurring Topics

Another aspect of this protocol that tells us something about S3's problem solving processes are the topics that recur throughout the protocol. She will bring up an issue, explore it for a while, sometimes even seeming to resolve it, then drop the subject for a while, even tens of minutes, only to bring it up again as new information is acquired that may be relevant. While she does this in a minor way in many places in the protocol, the two that permeate the whole design are the hyphen issue and the form of the term table data structure.

This suggests that S3's problem solving processes are very adept at accessing information whenever it is relevant. Moreover,

information is marked according to its relative importance and identified with that part of the design to which it is relevant.

Early in the protocol, S3 notices that the text may contain hyphens and that this complicates the comparison process. At this point, S3 only notes this "as being a problem when you come around to comparing" (249-250). This issue is not considered for long portions of the protocol, but it emerges whenever a module that is related to the compare operation or accessing the text is considered. After critiquing the final version of the design, S3 returns to the hyphen issue (1345-1367), reconsidering much of the information that was generated earlier, and considers other aspects of this issue that she would consider on further iterations of the design.

S3's examination of hyphens clearly permeates the entire protocol. She is, however, able to ignore the issue for a much as a half hour, bringing it up again as soon as a related module is being considered. It is interesting that she is able to recall this issue at precisely the relevant moments, considering the number and complexity of other items that she has had to retain and deal with in the interval. Notice that no external notation about hyphens is made until quite late in the protocol.

This quite remarkable memory for the hyphen problem tells us several things about how S3 must organize the information she gathers in the course of expanding a design. First, such information must be marked according to its importance; S3 generates far too much data for

it all to be as easily accessible as the hyphen information was. Second, the information must be tagged in some way with the part of the design to which it is relevant. S3 never mentioned hyphens during the "read terms" expansion, but it was one of the first things mentioned when she came to expanding the "construct index" module.

Some additional ideas about the organization of S3's memory can be derived from the ways she handles the construction of the term table data structure. This issue is considered several times and each consideration involves a depth-first exploration of the corresponding goals. The goals considered at each iteration, however, differ. In the first segment of the protocol, she decides to make a "table of terms", with each entry containing a term and a pointer to a linked list of the page numbers on which the term appears. Initially, there is some confusion over whether this table should contain the actual page numbers or merely a count of the number of times the term occurs in the text

In the second segment of the protocol, S3 proceeds to construct a term table that contains a pointer and a count, except the pointer is a pointer to a string containing the term itself and the count is a count of the number of characters in the term (411-431). No mention is made of page numbers until much later in the protocol, when she has found a term that matches the text (1142-1147). She does, however, treat the string table as though it were a linked list, discussing routines for insertions and deletions of list items and for "garbage collection".

It appears as if she has forgotten most of the information that she generated about this data structure in the earlier segment. She apparently remembers only that it was to contain linked lists, pointers, and counts, and she appears to reconstruct a new data structure based on this information and the fact that, at this point, she is looking for a place to put the term she has just read.

'One way for S3 to manage the large amount of information she generates as she explores facets of the design would be to summarize the information and only retain the summary. Frequently, only one or two points from an extended episode will be relevant later in the design and, if more information is needed, it can be regenerated if the necessary main points are remembered. It looks as if S3 was using just such a summarization strategy, but in the new context, the summary was insufficient to cause the original material to be properly reconstructed. S3 fails to notice this, however, because the data structure she does construct is a reasonable one, even if it is apparently not the one she sketched out in the initial segment of the protocol.

A BRIEF LOOK AT A SECOND EXPERT PROTOCOL

In order to further evaluate some of the results suggested by S3's protocol, we are performing a similar analysis on the protocol obtained from a second expert subject (S2). The problem given to this subject, to design a system to produce a page-keyed indexing system, was identical to that given to S3. In this section, we will describe our preliminary analysis of the first phase of this protocol, in which the abstract plan for the design was formulated. In general, this analysis (see Appendix G) produces results that are very similar to those obtained from S3.

S2 begins his design by constructing a high-level representation of the components that will be included in the final design and their interrelations (see the first panel of Figure 9). The problem model underlying S2's design appears very similar to that of S3. S2's next step is to consider the formats of the input and output files, and he then produces the abstract design that describes how these files will be processed. Finally, the necessary iterations are added (see the last panel of Figure 9).

Much of the evidence for S3's problem solving processes comes from consideration of her policies. Recall that policies are used for the generation and evaluation of design alternatives. This is also true of the policies we consider to be used by S2 (see Figure 10). Like S3, S2 has a policy to "divide the problem into inputs, processes, and

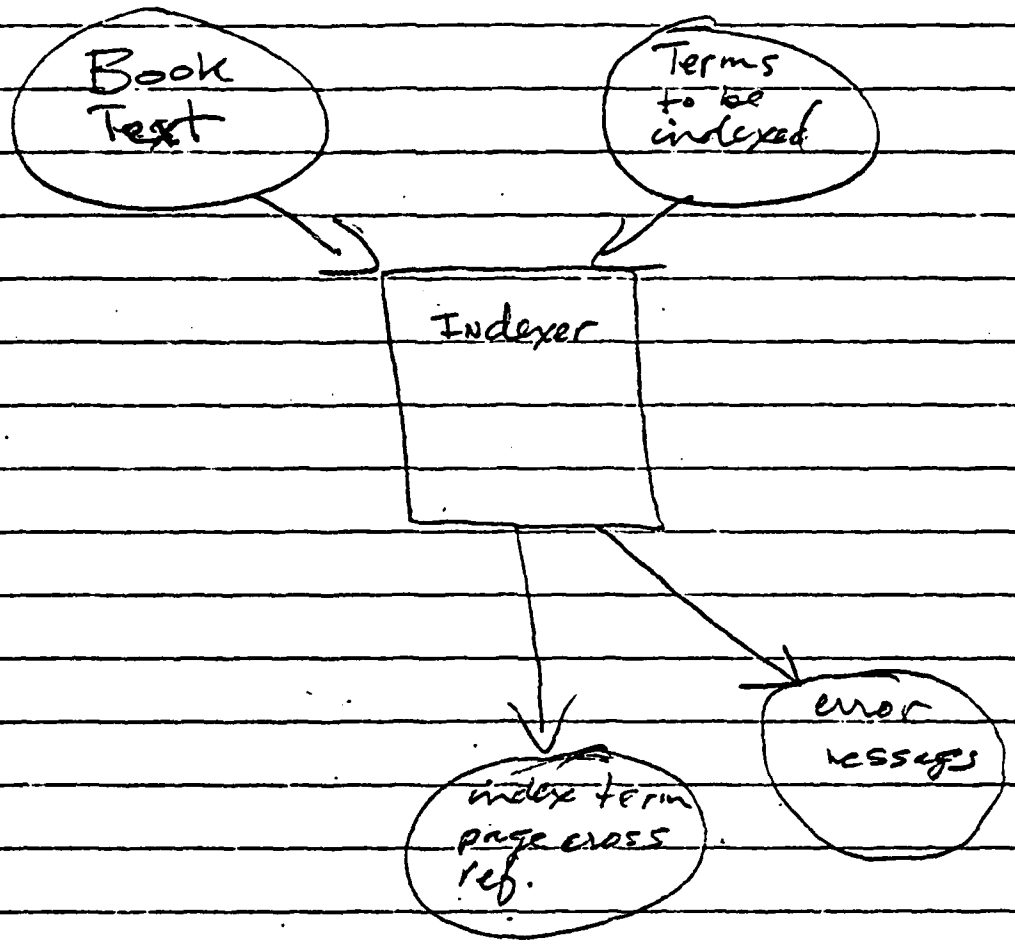


Figure 9. Subject S2 -- Abstract Design

(2) I/O Specs.

Book Text: - Ascii text character file
/ddddd is end-of-page
marker dddd is page #

- must input entire page
before indexing

Terms: 1 term per line
1 – 5 words long separated by
delimiter blks significant

Output XRef of Indexing:

indexed term	list of pages

ERROR: text file of printable messg.

Figure 9. Subject S2 -- Abstract Design
(continued)

(3)

(1.0) { read in terms to be indexed;
indexer: read page of book text;
~~set page #;~~
process page;

output xref;
}

(3.1) { initialize;
read in terms to be indexed;
while book text file not empty do
{ read page; ~~set page #;~~
process page;
}

output xref;
}

Figure 9. Subject S2 - Abstract Design
(continued)

divide problem into inputs, processes, and outputs

elaborate design in terms of implementation language primitives

make the most convenient assumption in ambiguous cases

use pseudo-programming language

use stepwise refinement

Figure 10. Subject S2 -- Policies

outputs", and in both protocols, this policy is used to identify the major components that will be required in the final design. S2's policy to "make the most convenient assumption in ambiguous cases" is similar to S3's policy to "choose the most convenient of multiple alternatives", and both are used for selecting among various design alternatives. While S3 used a policy to "expand top-down" to determine what element of the design to expand at a given time, S2 elected to "use stepwise refinement", which is very similar.

In summary, there is a great deal of similarity in the policies used by these two subjects. Although the exact form that these policies take differs somewhat, their basic use is in the generation and selection of design alternatives and in guiding the generally top-down expansion of the design.

The goals derived from our encoding of S2's protocol are contained in Appendix G. In comparison with those of S3, it can be noted that some (about 25 percent) are common to both subjects (see Figure 11). These common goals are concerned with the structure of the input files involved in the problem and with the issue of considering and then deferring the problem of hyphens appearing in the text. The remaining goals, however, are different, reflecting somewhat different approaches to the design task. For example, S2 constructs a separate file to store the results of the comparisons, while S3 stores these results in the term table data structure along with the terms.

S3: defer goal of resolving problem of hyphens having two functions
S2: defer goal to determine how to handle hyphens

S3: describe input files
S2: describe input files

S3: create problem model
S2: define problem in high-level terms

S3: resolve problem of hyphens having two functions
S2: determine how to handle hyphens

S3: describe term file
S2: describe term file

S3: describe text file
S2: describe text file

S3: expand FC (flowchart) to detailed level
S2: expand components of abstract design

S3: reread specifications for form of terms
S2: reread specifications for form of term file

S3: reread specifications fo text file
S2: reread specifications for form of text file

Figure 11. Goals Common to S3 and S2

	active	satisfied
1.	6, 4, 38	
7.	6, 4, 38, 31	
10.	6, 4, 38, 31, 17	
11.	6, 4, 38, 31, 3	17, 3
12.	6, 4, 38, 25	31
13.	6, 4	38, 25
14.	6, 4, 19, 21, 22	
20.	6, 4, 19, 21	22
21.	6, 4, 19, 21, 8, 9	
22.	6, 4, 19, 21, 8, 9, 13	
23.	6, 4, 19, 21, 8, 9, 13, 12	
24.	6, 4, 19, 21, 8, 9, 13, 12, 34	
26.	6, 4, 19, 21, 8, 9, 13, 12	34
31.	6, 4, 19, 21, 8, 9, 12	13
33.	6, 4, 19, 21, 8, 9, 12, 33	
35.	6, 4, 19, 21, 8, 9, 12	33
36.	6, 4, 19, 21, 8, 9, 12, 16	
37.	6, 4, 19, 21, 8, 9, 12, 16, 32	
38.	6, 4, 19, 21, 8, 9, 12, 16	32
39.	6, 4, 19, 21, 8, 9, 12	16
42.	6, 4, 19, 21, 8, 9, 12, 18	
43.	6, 4, 21, 9	19, 8, 12, 18
45.	6, 4, 21, 9, 14	
46.	6, 4, 21, 9, 14, 35	
47.	6, 4, 21, 9, 14	35

Figure 12. Subject S2 -- Example of Goal Stack

1-6. define problem in high-level terms
  6-19. determine inputs
    7-8. describe input files
      8-13. describe text file
        10-34. reread specifications for form of text file
      9-12. describe term file
        11-33. reread specifications for form of term file
        12-16. determine delimiters for words
          13-32. reread specifications for form of delimiters
        14-18. determine if blanks are significant
  6-21. determine outputs
    7-9. describe output files
      21-7. describe error message file
      15-14. describe xref (cross-reference file)
        17-39. use standard output format for xref
          18-11. describe standard output format
            19-23. determine what to do with multiple occurrences
              20-37. reread specifications to determine what to
                with multiple occurrences
        16-35. reread specifications for form of xref
1-38. restate problem in own terms
  2-31. reexamine specifications
  3-17. determine how to handle hyphens
    4-3. defer goal to determine how to handle hyphens
  5-25. determine where on line page number appears

Figure 13. Subject S2 -- Example of Goal Structure
(Note: first number indicates relative order -- second
identifies goal in "goal stack")

Although the specific goals involved differ, the patterns of goal activation and goal satisfaction are identical. That is, there are depth-first explorations of design components that have been identified as necessary. The goal stacks generated for each subject consist of a number of episodes (see Appendix D and Figure 6 for S3 and Appendix G and Figure 12 for S2), and increases in the number of active goals are due to a depth-first expansion of some topic (see Figure 13). In addition, it can be noted that the mean length of a given episode and the number of goals involved in a single episode are very similar for both subjects.

Although we have not examined the notes from S2 in the detail that we have considered those of S3, in both protocols these elements are used to store information that constrains the form and content of the developing design. The sources of this information also appears to be common for both S2 and S3.

In summary, although these two subjects adopted somewhat different approaches to this problem that resulted in somewhat different designs, there are commonalities in the use of policies and notes and in the dynamics of generating and maintaining a list of active goals. For both subjects, there are both breadth-first and depth-first expansions and other processes guide these top-down processes. We consider this to be a strong indication that there are also large commonalities in the problem solving processes that guide the design behavior of each subject.

DISCUSSION

In this report, have taken a detailed look at the steps a computer professional employs to solve a lengthy design problem. In this final section, we will review what we have learned about the processes that guide design behavior.

Breadth-First Expansions

First, we can note that S3 generates the design in a top-down manner. At a macro-level, the level of the design elements included in the flowchart, it is also primarily breadth-first. Her first iteration, the problem model, is an abstract decomposition of the problem, which is probably constructed soon after reading the specifications. This is expanded during the first third of the protocol into her abstract design. The remainder of the problem solving session is concerned with the expansion of each of these boxes. From the discussion at the end of the protocol, it appears that S3 would, on her next pass at a solution to this problem, again take each box of the current flowchart in turn and further refine it.

At every iteration, S3 examines each element of the design in its proper (i.e., top-down, breadth-first) sequence in the solution. Only on rare occasions does she jump around in the solution, either forward or backward. She avoids backtracking because of her desire to minimize the possible interactions that could occur with other existing modules. In the one case where she examines a later module prematurely (the

decision that hyphens will be handled in the "compare text:term" module), she merely notes this constraint; she makes no attempt to further expand the module, even to determine if the assigned process is achievable.

S2 shows this pattern even more strongly. He takes his design through more iterations than S3, in the portion of the protocol not discussed in this paper, but each is a completely breadth-first expansion of the previous solution. In the segment of the protocol that we have analyzed, he shows no deviations from a purely sequential consideration of each module.

Depth-First Expansions

While both protocols are strictly breadth-first when considered at a macro-level, when examined at the more micro-level of individual goals, both subjects exhibit a strong tendency to expand the solution depth-first. A subproblem is posed, a goal to solve it is set up, subgoals and sub-subgoals are created and resolved as needed. Only then is the next subproblem attempted. The few exceptions to this, where a set of sibling goals are successively enumerated, occur only under two circumstances. The first is at the top levels, where the subject is using some canonical set of goals that he or she presumably begins every such problem with (e.g., goals to describe both input and output files are enumerated breadth-first). The second occasion occurs when the current goal is to process some collection of items (e.g., multiple input files). The members of the collection are typically

contained in some external memory, and the subject enumerates each of the items to be processed before beginning detailed consideration of any of them. Not only are goals considered in a breadth-first sequence only under these restricted circumstances, but even when a set of goals is initially explored in this manner, each member of the set is then further expanded and resolved depth-first.

Stopping Rules and Iterations

When goals are expanded in a depth-first order, it is necessary to consider what the rules are that control the limits, or bounds, on the depth of the search; that is, what are the stopping rules for the subgoal expansion? From the fact that the design grows iteratively and breadth-first at the more global level, we must conclude that the designer is not considering each problem component until the final, ultimate level of resolution is reached; rather, the designer judges some level of resolution to be "appropriate" for a given iteration.

These subjects apparently have very little difficulty deciding when to terminate consideration of a goal. S3 shows uncertainty as to whether to continue expanding a goal on only one occasion; the first time the hyphen issue comes up, she grapples with the question of whether it is "important" enough to require resolution at this level. She vacillates, eventually deciding to put it off, but it is never clear what information she uses to make that decision. In the cases where the stopping decision causes her no trouble, the criteria she uses are even less obvious.

The stopping rules clearly vary with each iteration of the design. During the development of the abstract flowchart, S3 is primarily concerned with enumerating the "important" processes that must be contained in the solution. These include input file descriptions, internal data structures, and the major modules of the design. In the second iteration she is much more concerned with control structures, anomalous conditions, and some of the details of the design.

Two examples clearly illustrate the iterative nature of these stopping rules. First, consider how the "compare" routine is handled. During the development of the abstract design, it rates one sentence; after the term table is built, she notes that the terms and text are to be compared. No potential problems are even considered. The second half of the protocol, well over an hour of S3's time, is spent expanding this module, in particular dealing with its complex control structure and the many anomalous conditions which may occur.

The second example concerns the term table data structure. S3 spends approximately equal amounts of time dealing with this structure on each iteration of the design. However, she focuses on very different aspects of the term table each time. During the first iteration the general form of the data structure is at issue. She is particularly concerned with the matter of whether to store the page numbers for each term in an array or in a linked list. In the second pass she is more concerned with the details of the structure -- that it is to contain pointers to the terms, counts, and pointers to page

numbers. The data structure is still not completely defined; on some later iteration S3 must elaborate on the form of the pointers and counts and how the table is to be initialized.

The rules that control the bounds of this depth-first expansion are undoubtedly one of the most complex aspects of S3's design behavior. Similar rules are apparent in S2's protocol. These rules tend to insure that goals are not explored in either too much or too little detail, and they contribute to the relative ease or difficulty with which the design is expanded. At this point in our analysis we have no particular insights into what these stopping rules are. We suspect, however, that the sophistication of such rules may be one of the areas that separates experts from novices.

Constraints on Top-Down Expansions

As with any complex skill, top-down processes that guide expansion of the design (or problem solution) must interact with other processes for the design to be successfully produced. These processes operate to constrain the expansion of the top-down processes. In essence, they act as a sort of filter, limiting the possibilities to be considered by the designer as the solution is elaborated. We observe these processes operating primarily in two ways -- in the policies, and in some of the notes.

Many of the policies we observed, in particular those that allow designers to select among multiple alternatives, are used to channel the developing design in particular directions. In S3's protocol, for

example, "define returns from modules" operates on the perceptual features of the flowchart and gives her a systematic procedure for deciding in what order to expand the design. The policies to "be efficient", to "choose the most convenient alternative", and to "ask the experimenter" allow her to eliminate potential alternate solutions without the need to explore each possibility in detail. The policies to "consider anomalous conditions" and to "critique the design by considering alternative solutions" help S3 ensure the correctness of the design by generating plausible solutions to subproblems in cases where the first solution considered is likely to be incomplete or incorrect. In an earlier section, we noted the similarities between the policies of S3 and S2. Apparently, designers can use these "rules of thumb" to minimize the amount of effort needed to ensure that the design is expanded correctly at such points.

Other processes generate information that is also used to generate or select alternatives. Frequently, this information is generated by an active goal and is used to determine the subgoals necessary to perform that goal's function. At other times, information is generated that is not relevant to the subgoal currently being explored. Sometimes this datum is recalled and employed elsewhere in the design, often tens and occasionally hundreds of lines of the protocol later. Even more frequently, however, the information is never used. This is both because many of the items generated are not relevant to any later module, and because S3's memory for her notes is less than perfect.

Often, however, when beginning a new topic, S3 is able to retrieve out of the large amount of information she has generated a note that is relevant to the solution of her current subgoal.

Summary

The problem solving processes we have described enable a designer to generate the goal structures underlying the solution of a design task. Our analysis has focussed on the goals employed by designers as well as the information generated in the course of producing a solution. Our subjects solved a large number of subproblems while producing their designs. The goal information was apparently developed as a means of keeping track of what subproblem was being worked on and how it related to other pieces of the solution.

In a previous section, we reviewed other research using complex tasks. The problems used in such research, however, typically decomposed into two or three subcomponents. Both the subject and the theorist could maintain all these pieces in working memory concurrently; thus, most of these models assume that a plan was developed directly that covered all portions of the decomposition. Our more complex problem may have caused subjects to verbalize many more of the intermediate steps in their solutions, making the components which were considered and the interrelationships more apparent.

At a general level, there are some similarities between the processes described here and those incorporated by Hayes-Roth and Hayes-Roth (1979) in their Hearsay-based (see Lesser, Fennell, Erman,

and Reddy, 1975) model of errand planning behavior. Both studies focus on the processes involved in constructing solutions in complex tasks and both observe these processes operating in a multi-directional manner.

A basic difference, however, is that we observe a highly structured expansion, while Hayes-Roth and Hayes-Roth observe more "opportunistic" generation. This difference may be due to the fact that we are investigating design behavior, and for experienced designers such as those reported here, designs must be presented as hierarchical structures. This may cause experts to use highly structured generation processes. Such processes may also be due to memory and resource allocation heuristics that are used by the expert, but not known to novices. A second source of difference is that errand planning occurs at a fairly low level of detail while the designs considered here are fairly abstract.

REFERENCES

Bhaskar, R., & Simon, H.A. Problem solving in semantically rich domains: An example from engineering thermodynamics. **Cognitive Science**, 1977, 1, 192 215.

Chase, W.G., & Chi, M.T.H. **Cognitive skill: Implications for spatial skill in large-scale environments** (Technical Report No. 1). Pittsburgh, Pennsylvania: University of Pittsburgh, Learning Research and Development Center, December 1979.

Ernst, G.W., & Newell, A. **GPS: A case study in generality and problem solving.** New York: Academic Press, 1969.

Greeno, J.G. Natures of problem solving abilities. In W.K. Estes (Ed.), **Handbook of learning and cognitive processes.** Hillsdale, New Jersey, 1978, 239-270.

Greeno, J.G., Magone, M.E., & Chaiklin, S. Theory of constructions and set in problem solving. **Memory and Cognition**, 1979, **7**, 445-461.

Hayes-Roth, B., & Hayes-Roth, F. A cognitive model of planning. **Cognitive Science**, 1979, **3**, 275-310.

Larkin, J.H. **Skilled problem solving in physics: A hierarchical planning model** (Unpublished manuscript). Berkeley, California: University of California at Berkeley, September 1977.

Lesser, V.R., Fennell, R.D., Erman, L.D., & Reddy, D.R. Organization of the Hearsay-II speech understanding system. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, 1975, **ASSP-23**, 11-23.

Newell, A., & Simon, H.A. **Human problem solving.** Englewood Cliffs, New Jersey: Prentice Hall, 1972.

Simon, H.A. The structure of ill-structured problems. **Artificial Intelligence**, 1973, **4**, 181-201.

Sussman, G.J. Electrical design: A problem for artificial intelligence research. **Proceedings of the International Joint Conference on Artificial Intelligence**, Cambridge, Massachusetts, 1977, 894-900.

Navy

1 Dr. Jack R. Borsting
Provost & Academic Dean
U.S. Naval Postgraduate School
Monterey, CA 93940

1 Dr. Robert Breaux
Code N-711
NAVTRAEQUIPCEN
Orlando, FL 32813

1 Dr. Richard Elster
Department of Administrative Sciences
Naval Postgraduate School
Monterey, CA 93940

1 DR. PAT FEDERICO
NAVY PERSONNEL R&D CENTER
SAN DIEGO, CA 92152

1 Dr. John Ford
Navy Personnel R&D Center
San Diego, CA 92152

1 LT Steven D. Harris, MSC, USN
Code 6021
Naval Air Development Center
Warminster, Pennsylvania 18974

1 Dr. Patrick R. Harrison
Psychology Course Director
LEADERSHIP & LAW DEPT. (7b)
DIV. OF PROFESSIONAL DEVELOPMMENT
U.S. NAVAL ACADEMY
ANNAPOLIS, MD 21402

1 Dr. Norman J. Kerr
Chief of Naval Technical Training
Naval Air Station Memphis (75)
Millington, TN 38054

1 Dr. Leonard Kroeker
Navy Personnel R&D Center
San Diego, CA 92152

1 Dr. William L. Maloy
Principal Civilian Advisor for
Education and Training
Naval Training Command, Code 00A
Pensacola, FL 32508

Navy

1 Dr. Kneale Marshall
Scientific Advisor to DCNO(MPT)
OP01T
Washington DC 20370

1 CAPT Richard L. Martin, USN
Prospective Commanding Officer
USS Carl Vinson (CVN-70)
Newport News Shipbuilding and Drydock Co
Newport News, VA 23607

1 Dr William Montague
Navy Personnel R&D Center
San Diego, CA 92152

1 Commanding Officer
U.S. Naval Amphibious School
Coronado, CA 92155

1 Library
Naval Health Research Center
P. O. Box 85122
San Diego, CA 92138

1 Naval Medical R&D Command
Code 44
National Naval Medical Center
Bethesda, MD 20014

1 Ted M. I. Yellen
Technical Information Office, Code 201
NAVY PERSONNEL R&D CENTER
SAN DIEGO, CA 92152

1 Library, Code P201L
Navy Personnel R&D Center
San Diego, CA 92152

5 Technical Director
Navy Personnel R&D Center
San Diego, CA 92152

6 Commanding Officer
Naval Research Laboratory
Code 2627
Washington, DC 20390

Navy

1 Psychologist
ONR Branch Office
Bldg 114, Section D
666 Summer Street
Boston, MA 02210

1 Psychologist
ONR Branch Office
536 S. Clark Street
Chicago, IL 60605

1 Office of Naval Research
Code 437
800 N. Quincy SStreet
Arlington, VA 22217

5 Personnel & Training Research Programs
(Code 458)
Office of Naval Research
Arlington, VA 22217

1 Psychologist
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101

1 Office of the Chief of Naval Operations
Research, Development, and Studies Branc
(OP-102)
Washington, DC 20350

1 Captain Donald F. Parker, USN
Commanding Officer
Navy Personnel R&D Center
San Diego, CA 92152

1 DR. RICHARD A. POLLAK
ACADEMIC COMPUTING CENTER
U.S. NAVAL ACADEMY
ANNAPOLIS, MD 21402

1 Dr. Gary Poock
Operations Research Department
Code 55PK
Naval Postgraduate School
Monterey, CA 93940

Navy

1 Roger W. Remington, Ph.D
Code L52
NAMRL
Pensacola, FL 32508

1 Dr. Worth Scanland
Chief of Naval Education and Training
Code N-5
NAS, Pensacola, FL 32508

1 Dr. Robert G. Smith
Office of Chief of Naval Operations
OP-987H
Washington, DC 20350

1 Dr. Alfred F. Smode
Training Analysis & Evaluation Group
(TAEG)
Dept. of the Navy
Orlando, FL 32813

1 Dr. Robert Wisher
Code 309
Navy Personnel R&D Center
San Diego, CA 92152

Army

1 Technical Director
U. S. Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

1 HQ USAREUE & 7th Army
ODCSOPS
USAAREUE Director of GED
APO New York 09403

1 Col Gary W. Bloedorn
US Army TRADOC Systems Analysis Activity
Attn: ATAA-TH
WSMR, NM 88002

1 DR. RALPH DUSEK
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 Dr. Beatrice J. Farr
Army Research Institute (PERI-OK)
5001 Eisenhower Avenue
Alexandria, VA 22333

1 Dr. Ed Johnson
Army Research Institute
5001 Eisenhower Blvd.
Alexandria, VA 22333

1 Dr. Michael Kaplan
U.S. ARMY RESEARCH INSTITUTE
5001 EISENHOWER AVENUE
ALEXANDRIA, VA 22333

1 Dr. Milton S. Katz
Training Technical Area
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

1 Director
U.S. Army Human Engineering Labs
Attn: DRXHE-DB
Aberdeen Proving Ground, MD 21005

Army

1 Dr. Harold F. O'Neil, Jr.
Attn: PERI-OK
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

1 Dr. Robert Sasmor
U. S. Army Research Institute for the Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

1 Commandant
US Army Institute of Administration
Attn: Dr. Sherrill
FT Benjamin Harrison, IN 46256

1 Dr. Joseph Ward
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Air Force

1 Dr. Earl A. Alluisi
HQ, AFHRL (AFSC)
Brooks AFB, TX 78235

1 Dr. Genevieve Haddad
Program Manager
Life Sciences Directorate
AFOSR
Bolling AFB, DC 20332

1 Research and Measurment Division
Research Branch, AFMPC/MPCYPR
Randolph AFB, TX 78148

1 Dr. Marty Rockway (AFHRL/TT)
Lowry AFB
Colorado 80230

2 3700 TCHTW/TTGH Stop 32
Sheppard AFB, TX 76311

1 Jack A. Thorpe, Maj., USAF
Naval War College
Providence, RI 02846

1 Brian K. Waters, Lt Col, USAF
Air War College (EDV)
Maxwell AFB, AL 36112

Marines

1 H. William Greenup
Education Advisor (E031)
Education Center, MCDEC
Quantico, VA 22134

1 Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217

1 DR. A.L. SLAFKOSKY
SCIENTIFIC ADVISOR (CODE RD-1)
HQ, U.S. MARINE CORPS
WASHINGTON, DC 20380

Other DoD

12 Defense Documentation Center
Cameron Station, Bldg. 5
Alexandria, VA 22314
Attn: TC

1 Dr. Craig I. Fields
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

1 Dr. Dexter Fletcher
ADVANCED RESEARCH PROJECTS AGENCY
1400 WILSON BLVD.
ARLINGTON, VA 22209

1 Director, Research and Data
OASD(MRA&L)
3B919, The Pentagon
Washington, DC 20301

1 Military Assistant for Training and
Personnel Technology
Office of the Under Secretary of Defense
for Research & Engineering
Room 3D129, The Pentagon
Washington, DC 20301

1 HEAD, SECTION ON MEDICAL EDUCATION
UNIFORMED SERVICES UNIV. OF THE
HEALTH SCIENCES
6917 ARLINGTON ROAD
BETHESDA, MD 20014

Civil Govt

1 Dr. Susan Chipman
Learning and Development
National Institute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. Joseph I. Lipson
SEDR W-638
National Science Foundation
Washington, DC 20550

1 Dr. John Mays
National Institute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. Arthur Melmed
National Intitute of Education
1200 19th Street NW
Washington, DC 20208

1 Dr. Andrew R. Molnar
Science Education Dev.
and Research
National Science Foundation
Washington, DC 20550

1 Personnel R&D Center
Office of Personnel Managment
1900 E Street NW
Washington, DC 20415

1 Dr. H. Wallace Sinaiko
Program Director
Manpower Research and Advisory Services
Smithsonian Institution
801 North Pitt Street
Alexandria, VA 22314

1 Dr. Frank Withrow
U. S. Office of Education
400 Maryland Ave. SW
Washington, DC 20202

1 Dr. Joseph L. Young, Director
Memory & Cognitive Processes
National Science Foundation
Washington, DC 20550

Non Govt

1 Dr. John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

1 Anderson, Thomas H., Ph.D.
Center for the Study of Reading
174 Children's Research Center
51 Gerty Drive
Champiagn, IL 61820

1 Dr. John Annett
Department of Psychology
University of Warwick
Coventry CV4 7AL
ENGLAND

1 1 psychological research unit
Dept. of Defense (Army Office)
Campbell Park Offices
Canberra ACT 2600, Australia

1 Dr. R. A. Avner
University of Illinois
Computer-Based Educational Research Lab
Urbana, IL 61801

1 Dr. Alan Baddeley
Medical Research Council
Applied Psychology Unit
15 Chaucer Road
Cambridge CB2 2EF
ENGLAND

1 Dr. Patricia Baggett
Department of Psychology
University of Denver
University Park
Denver, CO 80208

1 Mr Avron Barr
Department of Computer Science
Stanford University
Stanford, CA 94305

Non Govt

1 Dr. Nicholas A. Bond
Dept. of Psychology
Sacramento State College
600 Jay Street
Sacramento, CA 95819

1 Dr. Lyle Bourne
Department of Psychology
University of Colorado
Boulder, CO 80309

1 Dr. Kenneth Bowles
Institute for Information Sciences
University of California at San Diego
La Jolla, CA 92037

1 Dr. Robert Brennan
American College Testing Programs
P. O. Box 168
Iowa City, IA 52240

1 Dr. John S. Brown
XEROX Palo Alto Research Center
3333 Coyote Road
Palo Alto, CA 94304

1 Dr. Bruce Buchanan
Department of Computer Science
Stanford University
Stanford, CA 94305

1 DR. C. VICTOR BUNDERSON
WICAT INC.
UNIVERSITY PLAZA, SUITE 10
1160 SO. STATE ST.
OREM, UT 84057

1 Charles Myers Library
Livingstone House
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

1 Dr. William Chase
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

Non Govt

1 Dr. Micheline Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

1 Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, Ma 02138

1 Dr. Meredith P. Crawford
American Psychological Association
1200 17th Street, N.W.
Washington, DC 20036

1 Dr. Kenneth B. Cross
Anacapa Sciences, Inc.
P.O. Drawer Q
Santa Barbara, CA 93102

1 Dr. Hubert Dreyfus
Department of Philosophy
University of California
Berkely, CA 94720

1 LCOL J. C. Eggenberger
DIRECTORATE OF PERSONNEL APPLIED RESEARC
NATIONAL DEFENCE HQ
101 COLONEL BY DRIVE
OTTAWA, CANADA K1A OK2

1 Dr. Ed Feigenbaum
Department of Computer Science
Stanford University
Stanford, CA 94305

1 Mr. Wallace Feurzeig
Bolt Beranek & Newman, Inc.
50 Moulton St.
Cambridge, MA 02138

1 Dr. Victor Fields
Dept. of Psychology
Montgomery College
Rockville, MD 20850

Non Govt

1 Dr. Edwin A. Fleishman
Advanced Research Resources Organ.
Suite 900
4330 East West Highway
Washington, DC 20014

1 DR. JOHN D. FOLLEY JR.
APPLIED SCIENCES ASSOCIATES INC
VALENCIA, PA 16059

1 Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

1 Dr. Alinda Friedman
Department of Psychology
University of Alberta
Edmonton, Alberta
CANADA T6G 2E9

1 Dr. R. Edward Geiselman
Department of Psychology
University of California
Los Angeles, CA 90024

1 DR. ROBERT GLASER
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

1 Dr. Marvin D. Glock
217 Stone Hall
Cornell University
Ithaca, NY 14853

1 Dr. Daniel Gopher
Industrial & Management Engineering
Technion-Israel Institute of Technology
Haifa
ISRAEL

1 DR. JAMES G. GREENO
LRDC
UNIVERSITY OF PITTSBURGH
3939 O'HARA STREET
PITTSBURGH, PA 15213

Non Govt

1 Dr. Ron Hambleton
School of Education
University of Massechusetts
Amherst, MA 01002

1 Dr. Harold Hawkins
Department of Psychology
University of Oregon
Eugene OR 97403

1 Dr. Barbara Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

1 Dr. Frederick Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

1 Dr. Dustin H. Heuston
Wicat, Inc.
Box 986
Orem, UT 84057

1 Dr. James R. Hoffman
Department of Psychology
University of Delaware
Newark, DE 19711

1 Glenda Greenwald, Ed.
"Human Intelligence Newsletter"
P. O. Box 1163
Birmingham, MI 48012

1 Dr. Earl Hunt
Dept. of Psychology
University of Washington
Seattle, WA 98105

3 Journal Supplement Abstract Service
American Psychological Association
1200 17th Street N.W.
Washington, DC 20036

Non Govt

1 Dr. Wilson A. Judd
McDonnell-Douglas
Astronautics Co.-St. Louis
P.O. Box 30204
Lowry AFB, CO 80230

1 Dr. Steven W. Keele
Dept. of Psychology
University of Oregon
Eugene, OR 97403

1 Dr. Walter Kintsch
Department of Psychology
University of Colorado
Boulder, CO 80302

1 Dr. David Kieras
Department of Psychology
University of Arizona
Tuscon, AZ 85721

1 Dr. Stephen Kosslyn
Harvard University
Department of Psychology
33 Kirkland Street
Cambridge, MA 02138

1 Mr. Marlin Kroger
1117 Via Goleta
Palos Verdes Estates, CA 90274

1 Dr. Jill Larkin
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

1 Dr. Alan Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

1 Dr. Michael Levine
210 Education Building
University of Illinois
Champaign, IL 61820

Non Govt

1 Dr. Robert A. Levit
Director, Behavioral Sciences
The BDM Corporation
7915 Jones Branch Drive
McClean, VA 22101

1 Dr. Charles Lewis
Faculteit Sociale Wetenschappen
Rijksuniversiteit Groningen
Oude Boteringestraat
Groningen
NETHERLANDS

1 Dr. Robert R. Mackie
Human Factors Research, Inc.
5775 Dawson Avenue
Goleta, CA 93017

1 Dr. Mark Miller
Computer Science Laboratory
Texas Instruments, Inc.
Mail Station 371, P.O. Box 225936
Dallas, TX 75265

1 Dr. Allen Munro
Behavioral Technology Laboratories
1845 Elena Ave., Fourth Floor
Redondo Beach, CA 90277

1 Dr. Donald A Norman
Dept. of Psychology C-009
Univ. of California, San Diego
La Jolla, CA 92093

1 Dr. Seymour A. Papert
Massachusetts Institute of Technology
Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

1 Dr. James A. Paulson
Portland State University
P.O. Box 751
Portland, OR 97207

1 MR. LUIGI PETRULLO
2431 N. EDGEWOOD STREET
ARLINGTON, VA 22207

Non Govt

1 DR. PETER POLSON
DEPT. OF PSYCHOLOGY
UNIVERSITY OF COLORADO
BOULDER, CO 80309

1 DR. DIANE M. RAMSEY-KLEE
R-K RESEARCH & SYSTEM DESIGN
3947 RIDGEMONT DRIVE
MALIBU, CA 90265

1 Dr. Fred Reif
SESAME
c/o Physics Department
University of California
Berkely, CA 94720

1 Dr. Ernst Z. Rothkopf
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

1 Dr. David Rumelhart
Center for Human Information Processing
Univ. of California, San Diego
La Jolla, CA 92093

1 DR. WALTER SCHNEIDER
DEPT. OF PSYCHOLOGY
UNIVERSITY OF ILLINOIS
CHAMPAIGN, IL 61820

1 Dr. Alan Schoenfeld
Department of Mathematics
Hamilton College
Clinton, NY 13323

1 DR. ROBERT J. SEIDEL
INSTRUCTIONAL TECHNOLOGY GROUP
HUMRRO
300 N. WASHINGTON ST.
ALEXANDRIA, VA 22314

1 Committee on Cognitive Research
% Dr. Lonnie R. Sherrod
Social Science Research Council
605 Third Avenue
New York, NY 10016

Non Govt

1 Robert S. Siegler
Associate Professor
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

1 Dr. Robert Smith
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

1 Dr. Richard Snow
School of Education
Stanford University
Stanford, CA 94305

1 Dr. Kathryn T. Spoehr
Department of Psychology
Brown University
Providence, RI 02912

1 Dr. Robert Sternberg
Dept. of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

1 DR. ALBERT STEVENS
BOLT BERANEK & NEWMAN, INC.
50 MOULTON STREET
CAMBRIDGE, MA 02138

1 Dr. David Stone
ED 236
SUNY, Albany
Albany, NY 12222

1 DR. PATRICK SUPPES
INSTITUTE FOR MATHEMATICAL STUDIES IN
THE SOCIAL SCIENCES
STANFORD UNIVERSITY
STANFORD, CA 94305

Non Govt

1 Dr. Kikumi Tatsuoka
Computer Based Education Research
Laboratory
252 Engineering Research Laboratory
University of Illinois
Urbana, IL 61801

1 Dr. John Thomas
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

1 DR. PERRY THORNDYKE
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA, CA 90406

1 Dr. Douglas Towne
Univ. of So. California
Behavioral Technology Labs
1845 S. Elena Ave.
Redondo Beach, CA 90277

1 Dr. J. Uhlaner
Perceptronics, Inc.
6271 Variel Avenue
Woodland Hills, CA 91364

1 Dr. Benton J. Underwood
Dept. of Psychology
Northwestern University
Evanston, IL 60201

1 Dr. Willard S. Vaughan, Jr.
Oceanautics, Inc.
422 Sixth Street
Annapolis, MD 21403

1 Dr. Phyllis Weaver
Graduate School of Education
Harvard University
200 Larsen Hall, Appian Way
Cambridge, MA 02138

1 Dr. David J. Weiss
N660 Elliott Hall
University of Minnesota
75 E. River Road
Minneapolis, MN 55455

Non Govt

1 DR. GERSHON WELTMAN
PERCEPTRONICS INC.
6271 VARIEL AVE.
WOODLAND HILLS, CA 91367

1 Dr. Keith T. Wescourt
Information Sciences Dept.
The Rand Corporation
1700 Main St.

ACKNOWLEDGEMENT

Computer time was provided by the SUMEX-AIM computer facility under grant number RR-00785 from the National Institute of Health.